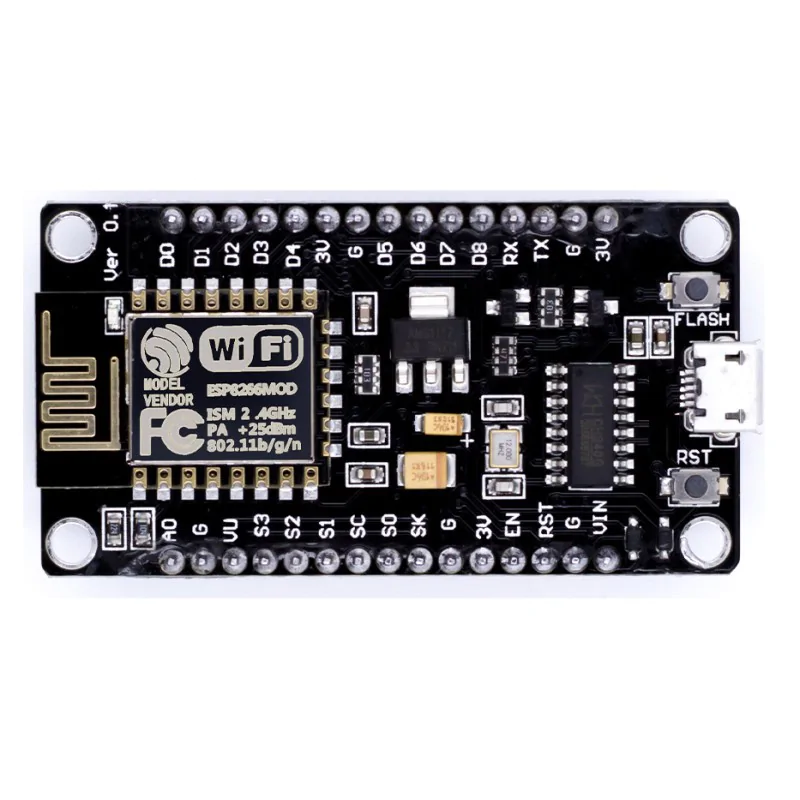
Projekt zdalnie sterowanej linii produkcyjnej

# Spis treści

1. wykorzystane moduły
2. sposób wykonywania
3. kody wykorzystane do realizacji projektu
4. zdjęcia
5. sposób działania finalnego

1.moduły

1. Modułem serce całego projektu jest płytka ESP8266

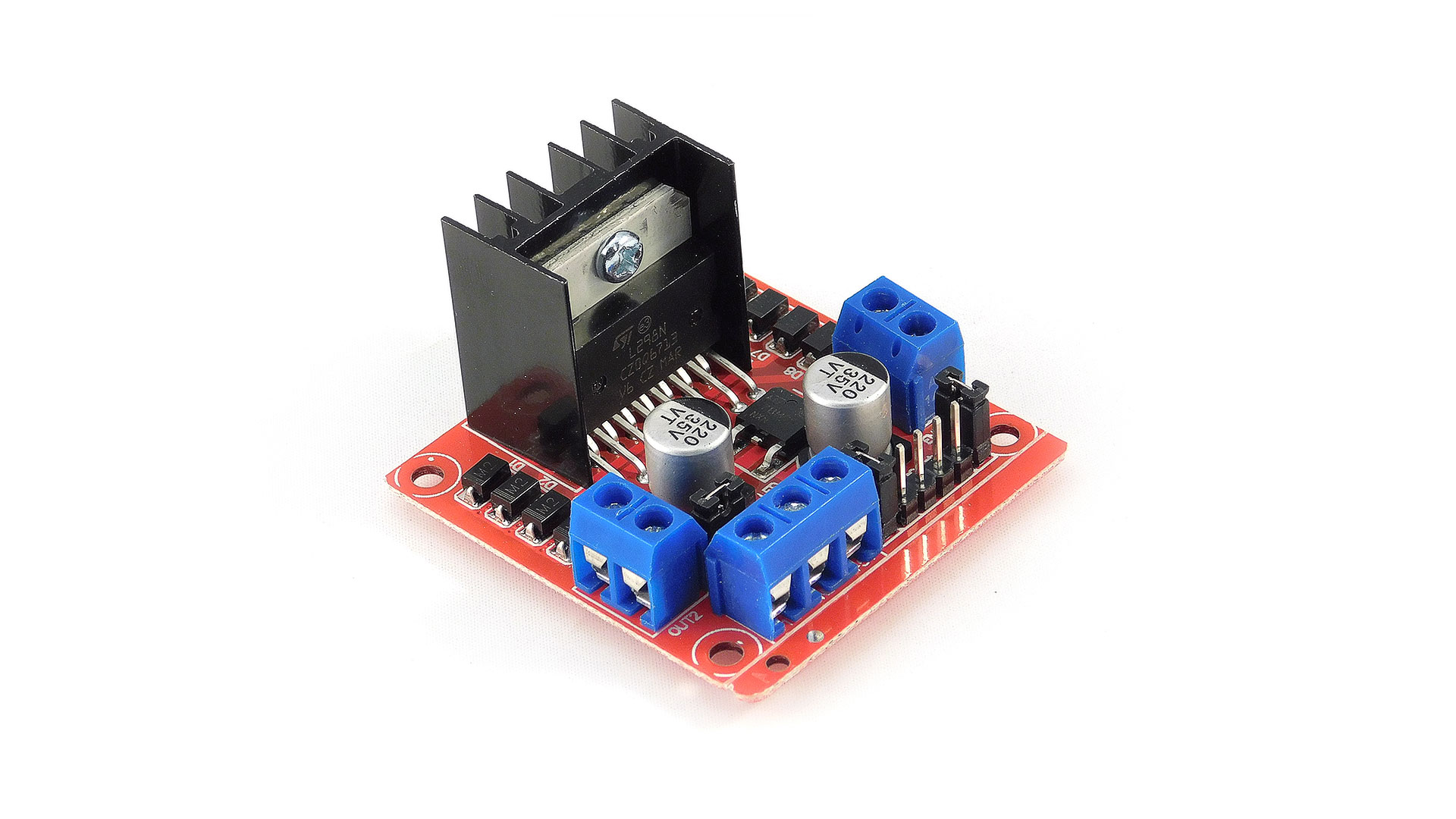


Moduł ten łączy z sobą wszystkie inne w celu przekazywania im informacji i zadań które mają wykonać. ESP8266 również posłużył do stworzenia serwera sieciowego z którego zdalnie jest sterowane urządzenie

2. RFID-RC522 jest to płytka która służy do zczytywania kart które mają dawać dostęp do strony w której jest przeprowadzane sterowanie silnikiem który napędza całą linie produkcyjną



3. Moduł L298N służyn nam do sterowania silnikiem oraz kontroli jego prędkości obrotów i zasilania silnika.



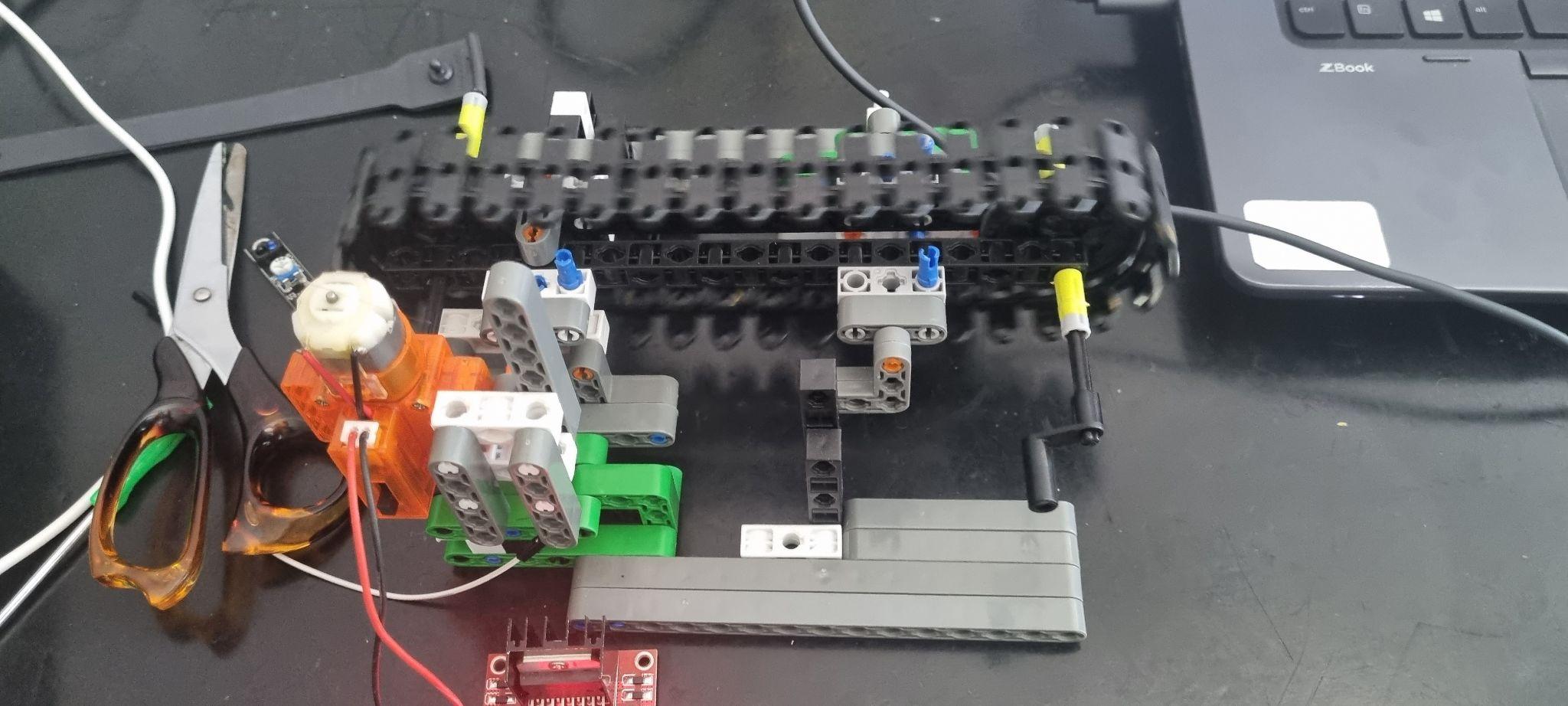
zasilacz 5V 2A służy nam do zasilenia modułu L298N

Przewody łączące

części konstruktorskie

2. Sposób wykonania

Rozpoczęto od budowy urządzenia które będzie obsługiwało silniki:



urządzenie stoi na stelażu który utrzymuje taśmociąg który jest wprawiany w ruch dzięki silnikowi

W późniejszym etapie silnik został podłączony do urządzenia.

Punktem środkowym było podłączenie wszystch potrzebnych portów L298N do silników (ENA=D1 , IN1=D2 IN2=D2 OUT1=SILNIK OUT2=SILNIK) zostało podpiętę zasilanie opisane powyżej.(Piny opisane w podpunkcie zostały odpowiednio podłączone do płytki ESP8266 według przepisów powyżej)

Następnym punktem było podpięcie systemu RFID-RC522(RFID jest zasilany napięciem nie większym niż 3.3V!) do ESP8266 w następującej kolejności (GND-GND 3.3V-3.3V RST-D1

SDA-D8 SCK-D5 MOSI-D6 MISO-D7)

3.Kod wykorzystany do realizacji projektu

Kod Rfid

#include <SPI.h>

#include <MFRC522.h>

#define SS\_PIN D8

#define RST\_PIN D0

MFRC522 mfrc522(SS\_PIN, RST\_PIN); // Create MFRC522 instance.

void setup()

{

Serial.begin(115200); // Initiate a serial communication

SPI.begin(); // Initiate SPI bus

mfrc522.PCD\_Init(); // Initiate MFRC522

Serial.println("Approximate your card to the reader...");

Serial.println();

}

void loop()

{

// wykrywa karte

if ( ! mfrc522.PICC\_IsNewCardPresent())

{

return;

}

// wybiera jedna z kart

if ( ! mfrc522.PICC\_ReadCardSerial())

{

return;

}

//Show UID on serial monitor

Serial.print("UID tag :");

String content= "";

byte letter;

for (byte i = 0; i < mfrc522.uid.size; i++)

{

Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");

Serial.print(mfrc522.uid.uidByte[i], HEX);

content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));

content.concat(String(mfrc522.uid.uidByte[i], HEX));

}

Serial.println();

Serial.print("Message : ");

content.toUpperCase();

if (content.substring(1) == "09 BD 65 57") //change here the UID of the card/cards that you want to give access

{

Serial.println("Authorized access");

Serial.println();

delay(3000);

}

else {

Serial.println(" Access denied");

delay(3000);

}

}

Powyższy kod wykorzystuje się do zaczytywania danych z kart dostępu.

Program do sterowania silnikami zdalnie za pomocą serwera internetowego

| Porgram działania 2 silników z możliwością sterowania przez serwer internetowy |  |
| --- | --- |
|  |  |
|  | //Biblioteki wykorzystane do połączenia się z ESP8266 |
|  | #include <ESP8266WiFi.h> |
|  | #include <ESPAsyncTCP.h> |
|  | #include <ESPAsyncWebServer.h> |
|  |  |
|  | //poniższe komenty umożliwiają nam dostęp do sieci WI-Fi |
|  | const char\* ssid = "login"; |
|  | const char\* password = "hasło"; |
|  |  |
|  | //Przypisanie odpowiednich pinów do płytki |
|  | int ENA\_pin = D7; |
|  | int ENB\_pin = D8; |
|  | int IN1 = D1; |
|  | int IN2 = D2; |
|  | int IN3 = D3; |
|  | int IN4 = D4; |
|  |  |
|  | //Zadeklarowanie zmiennych to odpowiedniego działania silników |
|  | String slider\_value = "0"; |
|  | String slider\_value2 = "0"; |
|  | const char\* input\_parameter = "value"; |
|  | const char\* input\_parameter2 = "value2"; |
|  |  |
|  |  |
|  | //łaczenie z siecią na porcie 80 |
|  | AsyncWebServer server(80); |
|  | ////kod HTML , ccs i javascrit który opisuje nam wygląd stroy oraz zasadę działania tej strony |
|  | const char index\_html[] PROGMEM = R"rawliteral( |
|  | <!DOCTYPE HTML><html> |
|  | <head> |
|  | <meta name="viewport" content="width=device-width, initial-scale=1"> |
|  | <title>DC Motors Speed Control Web Server</title> |
|  | <style> |
|  | html {font-family: Times New Roman; display: inline-block; text-align: center;} |
|  | h2 {font-size: 2.3rem;} |
|  | h4 {font-size: 1.8rem;} |
|  | p {font-size: 2.0rem;} |
|  | body {max-width: 400px; margin:0px auto; padding-bottom: 25px;} |
|  | .slider { -webkit-appearance: none; margin: 14px; width: 360px; height: 25px; background: #FF0000; |
|  | outline: none; -webkit-transition: .2s; transition: opacity .2s;} |
|  | .slider::-webkit-slider-thumb {-webkit-appearance: none; appearance: none; width: 35px; height: 35px; background:#01070a; cursor: pointer;} |
|  | .slider::-moz-range-thumb { width: 35px; height: 35px; background: #01070a; cursor: pointer; } |
|  |  |
|  | .slider2 { -webkit-appearance: none; margin: 14px; width: 360px; height: 25px; background: #FF0000; |
|  | outline: none; -webkit-transition: .2s; transition: opacity .2s;} |
|  | .slider2::-webkit-slider-thumb {-webkit-appearance: none; appearance: none; width: 35px; height: 35px; background:#01070a; cursor: pointer;} |
|  | .slider2::-moz-range-thumb { width: 35px; height: 35px; background: #01070a; cursor: pointer; } |
|  | </style> |
|  | </head> |
|  | <body> |
|  | <h2>DC Motor Speed Control Web Server</h2> |
|  | <h4>Silnik 1</h4> |
|  | <p><span id="textslider\_value">%SLIDERVALUE%</span></p> |
|  | <p><input type="range" onchange="updateSliderPWM(this)" id="pwmSlider" min="0" max="255" value="%SLIDERVALUE%" step="1" class="slider"></p> |
|  |  |
|  | <br/> |
|  | <h4>Silnik 2</h4> |
|  | <p><span id="textslider\_value2">%SLIDERVALUE2%</span></p> |
|  | <p><input type="range" onchange="updateSliderPWM2(this)" id="pwmSlider2" min="0" max="255" value="%SLIDERVALUE2%" step="1" class="slider2"></p> |
|  |  |
|  | <script> |
|  | function updateSliderPWM(element) { |
|  | var slider\_value = document.getElementById("pwmSlider").value; |
|  | document.getElementById("textslider\_value").innerHTML = slider\_value; |
|  | console.log(slider\_value); |
|  | var xhr = new XMLHttpRequest(); |
|  | xhr.open("GET", "/slider?value="+slider\_value, true); |
|  | xhr.send(); |
|  |  |
|  | } |
|  | function updateSliderPWM2(element) { |
|  | var slider\_value2 = document.getElementById("pwmSlider2").value; |
|  | document.getElementById("textslider\_value2").innerHTML = slider\_value2; |
|  | console.log(slider\_value2); |
|  | var xhr2 = new XMLHttpRequest(); |
|  | xhr2.open("GET", "/slider?value2="+slider\_value2, true); |
|  | xhr2.send(); |
|  | } |
|  |  |
|  | </script> |
|  | </body> |
|  | </html> |
|  | )rawliteral"; |
|  |  |
|  | // Przypisywanie wartości suwaka do zmiennej |
|  | String processor(const String& var){ |
|  | if (var == "SLIDERVALUE"){ |
|  | return slider\_value; |
|  | }else if(var=="SLIDERVALUE2"){ |
|  | return slider\_value2; |
|  | }else{ |
|  | return String(); |
|  | } |
|  | } |
|  |  |
|  | //Zasada działania silników |
|  | void setup(){ |
|  | Serial.begin(115200); |
|  | delay(1000); |
|  | pinMode(ENA\_pin, OUTPUT); |
|  | pinMode(IN1, OUTPUT); |
|  | pinMode(IN2, OUTPUT); |
|  | pinMode(ENB\_pin, OUTPUT); |
|  | pinMode(IN3, OUTPUT); |
|  | pinMode(IN4, OUTPUT); |
|  |  |
|  | analogWrite(ENA\_pin, slider\_value.toInt()); |
|  | digitalWrite(IN1, LOW); |
|  | digitalWrite(IN2, LOW); |
|  | analogWrite(ENB\_pin, slider\_value2.toInt()); |
|  | digitalWrite(IN3, LOW); |
|  | digitalWrite(IN4, LOW); |
|  |  |
|  | WiFi.begin(ssid, password); |
|  | while (WiFi.status() != WL\_CONNECTED) { |
|  | delay(1000); |
|  | Serial.println("Connecting..."); |
|  | } |
|  |  |
|  | Serial.println(WiFi.localIP()); |
|  |  |
|  | server.on("/", HTTP\_GET, [](AsyncWebServerRequest \*request){ |
|  | request->send\_P(200, "text/html", index\_html, processor); |
|  | }); |
|  |  |
|  | server.on("/slider", HTTP\_GET, [] (AsyncWebServerRequest \*request) { |
|  | String message; |
|  | if (request->hasParam(input\_parameter)) { |
|  | message = request->getParam(input\_parameter)->value(); |
|  | slider\_value = message; |
|  | analogWrite(ENA\_pin, slider\_value.toInt()); |
|  | digitalWrite(IN1, HIGH); |
|  | digitalWrite(IN2, LOW); |
|  | }else if (request->hasParam(input\_parameter2)) { |
|  | message = request->getParam(input\_parameter2)->value(); |
|  | slider\_value2 = message; |
|  | analogWrite(ENB\_pin, slider\_value2.toInt()); |
|  | digitalWrite(IN3, HIGH); |
|  | digitalWrite(IN4, LOW); |
|  | }else{ |
|  | message = "No message sent"; |
|  | } |
|  | Serial.println(message); |
|  | request->send(200, "text/plain", "OK"); |
|  | }); |
|  |  |
|  | server.begin(); |
|  | } |
|  | void loop() { |
|  |  |
|  | } |

Powyższy program wczytuje informacje dotyczące sieci w późniejszym czasie tworzy odpowiednio stworzony web-server oraz wysterowywuje silnikami.

Sposób działania systemów

plik znajduje się w projekcie github